Analisis Kluster (Clustering) - DBSCAN

# Konsep Dasar

DBSCAN (Density-Based Spatial Clustering of Application with Noise) adalah algoritma clustering berbasis kepadatan yang dapat menemukan cluster dengan bentuk sembarang dan mengidentifikasi outlier (noise).

# Cara Kerja DBSCAN:

1. **Menemukan Parameter:**

* eps: Jarak maksimum antara dua titik data agar dianggap sebagai tetangga.
* min\_samples: Jumlah minimum titik data dalam suatu wilayah agar dianggap sebagai cluster.

1. **Mencari Core Points:** Titik data yang memiliki setidaknya min\_samples tetangga dalam eps disebut core points.
2. **Membentuk Cluster:** Core points yang saling bertetanggaan (dalam jarak eps) digabungkan menjadi satu cluster. Titik-titik data yang bukan core points tetapi berada dalam jarak eps dari core point disebut border points dan juga dimasukkan ke dalam cluster. Titik-titik data yang tidak termasuk dalam kategori core points atau border disebut noise points.

**Contoh Kode (Scikit-Learn)**

from sklearn.cluster import DBSCAN

# ... (kode untuk membaca dan mempersiapkan dataset)

# Membuat model DBSCAN

dbscan = DBSCAN(eps=0.5, min\_samples=5) # Contoh parameter

# Melatih model

labels = dbscan.fit\_predict(X)

**Tugas:**

1. Terapkan DBSCAN pada dataset “make\_moons” dari scikit-learn (dataset ini tidak dapat dipisahkan secara linear).
2. Visualisasikan hasil clustering dengan scatter plot, dengan warna titik berbeda untuk setiap cluster dan noise.
3. Coba beberapa kombinasi nilai eps dan min\_samples yang berbeda untuk melihat bagaimana hal tersebut mempengaruhi hasil clustering.

**Selamat Mengerjakan Tugas! 😀**

**Tugas Hari 38:**

1. **Terapkan DBSCAN:**

**import numpy as np**

**import pandas as pd**

**import matplotlib.pyplot as plt**

**from sklearn import datasets**

**from sklearn.cluster import DBSCAN**

**from sklearn.decomposition import PCA**

**from sklearn.metrics import silhouette\_score, adjusted\_rand\_score**

**# Load dataset iris**

**iris = datasets.load\_iris()**

**X = iris.data**

**y = iris.target**

**# Fungsi untuk menjalankan DBSCAN dengan parameter yang diberikan dan menampilkan hasilnya**

**def run\_dbscan(eps, min\_samples):**

**print(f"\nDBSCAN dengan eps={eps}, min\_samples={min\_samples}")**

**# Membuat model DBSCAN**

**dbscan = DBSCAN(eps=eps, min\_samples=min\_samples)**

**# Melatih model dan mendapatkan label cluster**

**labels = dbscan.fit\_predict(X)**

1. **Visualisasikan:**

**# Visualisasi Hasil DBSCAN dengan PCA**

**pca = PCA(2)**

**X\_pca = pca.fit\_transform(X)**

**plt.figure(figsize=(8, 6))**

**unique\_labels = set(labels)**

**colors = [plt.cm.Spectral(each) for each in np.linspace(0, 1, len(unique\_labels))]**

**for k, col in zip(unique\_labels, colors):**

**if k == -1:**

**col = [0, 0, 0, 1] # Hitam digunakan untuk noise.**

**class\_member\_mask = (labels == k)**

**xy = X\_pca[class\_member\_mask]**

**plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=tuple(col),**

**markeredgecolor='k', markersize=14)**

**plt.title(f'DBSCAN Clustering of Iris Dataset (2D PCA)\neps={eps}, min\_samples={min\_samples}')**

**plt.xlabel('Principal Component 1')**

**plt.ylabel('Principal Component 2')**

**plt.show()**

1. **Kombinasi:**

**# Kombinasi nilai eps dan min\_samples yang akan dicoba**

**eps\_values = [0.3, 0.5, 0.7]**

**min\_samples\_values = [3, 5, 7]**

**# Mencoba kombinasi nilai eps dan min\_samples**

**for eps in eps\_values:**

**for min\_samples in min\_samples\_values:**

**run\_dbscan(eps, min\_samples)**